SENAI Serviço Nacional de Aprendizagem Industrial

SQL

Structured Query Language (**SQL** – Linguagem de Consulta Estruturada) é uma linguagem da informática destinada a armazenar, manipular e obter dados armazenados em bases de dados relacionais;

- Surgiu em 1974 dentro da empresa IBM;

SENAI Serviço Nacional de Aprendizagem Industrial

SGBD escolhido: MySQL

OBS: Para instalar o MySQL vamos usar a ferramentas XAMP (Já trás em conjunto o MySQL (SGBD) e Apache (Servidor Web).

XAMPP

Tipo de dados

- Integer – Números inteiros
- Char(tamanho) – Caracteres com ocupação total
- Varchar(tamanho) – Caracteres com ocupação pacial
- Numeric(tamanho,decimais) – Define números reais com precisão decimal

  OBS: Numeric = Decimal
- Date - data
- Time - hora
- Timestamp – Data e hora juntos

SENAI Serviço Nacional de Aprendizagem Industrial

Os comandos são agrupados em 5 grupos:

Comandos DDL

- CREATE Criar banco, tabela, índice, visão
- ALTER Alterar estrutura
- DROP Excluir definitivamente
- TRUNCATE Limpar dados da tabela
- RENAME Renomear tabela/objeto

Comandos DDL: Create

Cria tabelas, bancos de dados, índices, visões etc.

Sintaxe básica:

```
CREATE TABLE nome_tabela (
   coluna1   tipo_dado   [NOT NULL],
   coluna2   tipo_dado    [NOT NULL],
   ...
   PRIMARY KEY (coluna)
);
```

OBS: Iniciamente é necessário criar a base de dados CREATE Database nomedabase;

Comandos DDL: Create

Cria tabelas, bancos de dados, índices, visões etc.

```sql
CREATE TABLE alunos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100),
    idade INT
);
```

OBS: Iniciamente é necessário criar a base de dados CREATE Database nomedabase;

Comandos DDL: Alter

Altera a estrutura de uma tabela existente.

Adicionar coluna

```
ALTER TABLE alunos ADD email VARCHAR(150);
```

Alterar tipo

```
ALTER TABLE alunos MODIFY idade SMALLINT;
```

Renomear tabela

```
 ALTER TABLE alunos RENAME TO estudantes;
```

Renomear coluna

```
ALTER TABLE fabrica
   CHANGE COLUMN email e_mail varchar(100);
```

Comandos DDL: Drop

Exclui tabelas, bancos de dados ou outros objetos.

Apaga tabela alunos

```
DROP TABLE alunos;
```

Apaga base de dados escola

```
DROP DATABASE escola;
```

Comandos DDL: Truncate

Apaga **todos os dados** da tabela, mas **mantém a estrutura**.

Apaga dados de alunos

```
TRUNCATE TABLE alunos;
```

OBS: Muito mais rápido que DELETE, mas não pode ser desfeito.

Comandos DDL: Rename

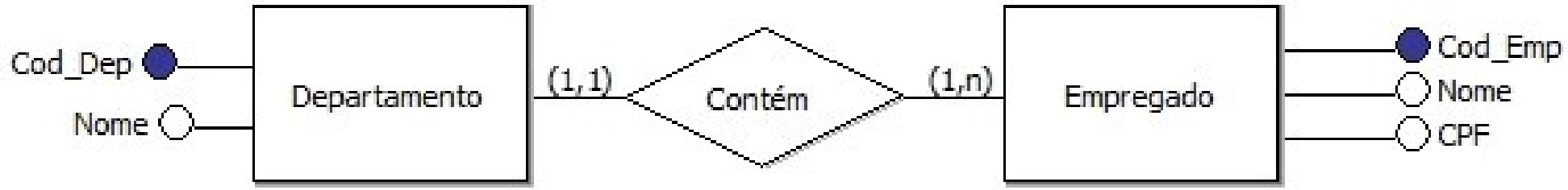
Renomeia objetos do banco (em alguns SGBDs é parte do ALTER).

Renomeia a tabela Alunos:

```
RENAME TABLE alunos TO estudantes;
```

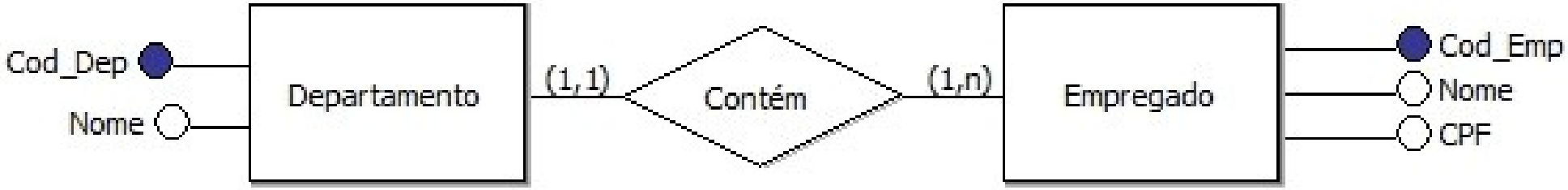
Chave Estrangeira

- Permite a relação entre tabelas em um banco dedados.



- Empregado vai receber a chave primária de Departamento.

Chave Estrangeira



Cod_Emp	Nome	CPF	Cod_dep
10	Marcos	024.700.234-21	01
11	Mateus	5555	02
12	Maria	7777	01

Cod_dep	Nome
01	TI
02	RH

SENAI Serviço Nacional de Aprendizagem Industrial

Chave Estrangeira

```
CREATE TABLE nome_tabela (
    coluna1   tipo_dado,
    coluna2   tipo_dado,
    ...
   PRIMARY KEY (coluna),
   FOREIGN KEY (coluna)
         REFERENCES nome_tabela (coluna_referenciada),
   UNIQUE (coluna)
);
```

Chave Estrangeira

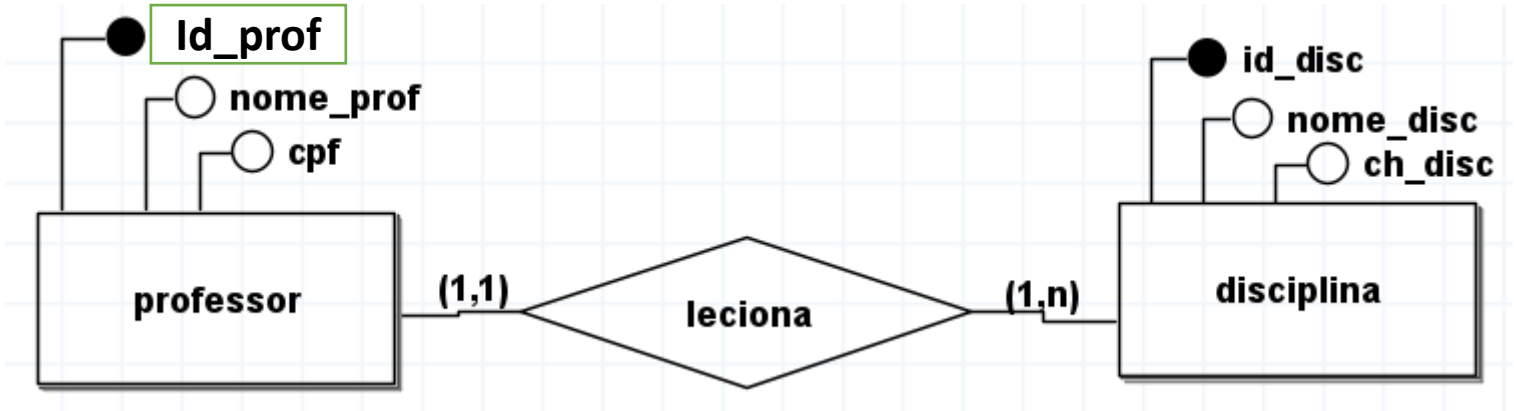
OBS: A tabela Departamento já tem que estar no BD, para que a coluna possa ser referenciada

```
CREATE TABLE Empregado (
    Cod_Emp   INT    NOT NULL    auto_increment,
    NOME      VARCHAR(40),
    CPF       INT,
    ID_Dep    INT,

   PRIMARY KEY (Cod_Emp),
   FOREIGN KEY (ID_Dep) REFERENCES Departamento (Cod_Dep),
   UNIQUE (CPF)
);
```

Exercício

- Crie as tabelas abaixo usando comandos SQL:



Comandos DML (Manipulação)

- INSERT – Insere registros na tabela
- UPDATE - Altera registros
- DELETE - Exclui registros

Comandos - **INSERT**

O comando **INSERT** adiciona novos dados em uma tabela.

sintaxe:

```
INSERT INTO nome_da_tabela (coluna1, coluna2, coluna3)
VALUES (valor1, valor2, valor3);
```

Exemplo – Inserindo dados na tabela professor:

```
INSERT INTO professor (id_prof, nome_prof, cpf)
VALUES (1, 'João Silva', '12345678900');
```

Id_prof	Nome_prof	cpf
1	João Silva	12345678900

Comandos - **INSERT**

O comando **INSERT** adiciona novos dados em uma tabela.

Exemplo – Inserindo dados na tabela disciplina:

```
INSERT INTO disciplina (id_disc, nome_disc, ch_disc,id_prof)
VALUES (10, 'Banco de Dados', 80,1);
```

Id_disc	Nome_disc	Ch_disc	Id_prof
10	Banco de Dados	80	1

Comandos - **UPDATE**

Modifica informações de um ou mais registros existentes em uma tabela.

Exemplo – Atualizando o nome de um professor:

Cuidado: Se o comando `WHERE` for omitido, **todos os registros** da tabela serão alterados.

```
UPDATE professor
SET nome_prof = ‘Marcos Silva'
WHERE id_prof = 1;
```

Id_prof	Nome_prof	cpf
1	João Silva	12345678900

Id_prof	Nome_prof	cpf
1	Marcos Silva	12345678900

Comandos - **DELETE**

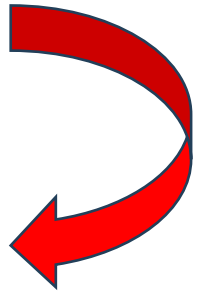
Modifica informações de um ou mais registros existentes em uma tabela.

Exemplo – Apagando professor:

Cuidado: Se o comando `WHERE` for omitido, **todos os registros** da tabela serão alterados.

```
DELETE FROM professor
WHERE id_professor = 1;
```

Id_prof	Nome_prof	cpf
1	Marcos Silva	12345678900

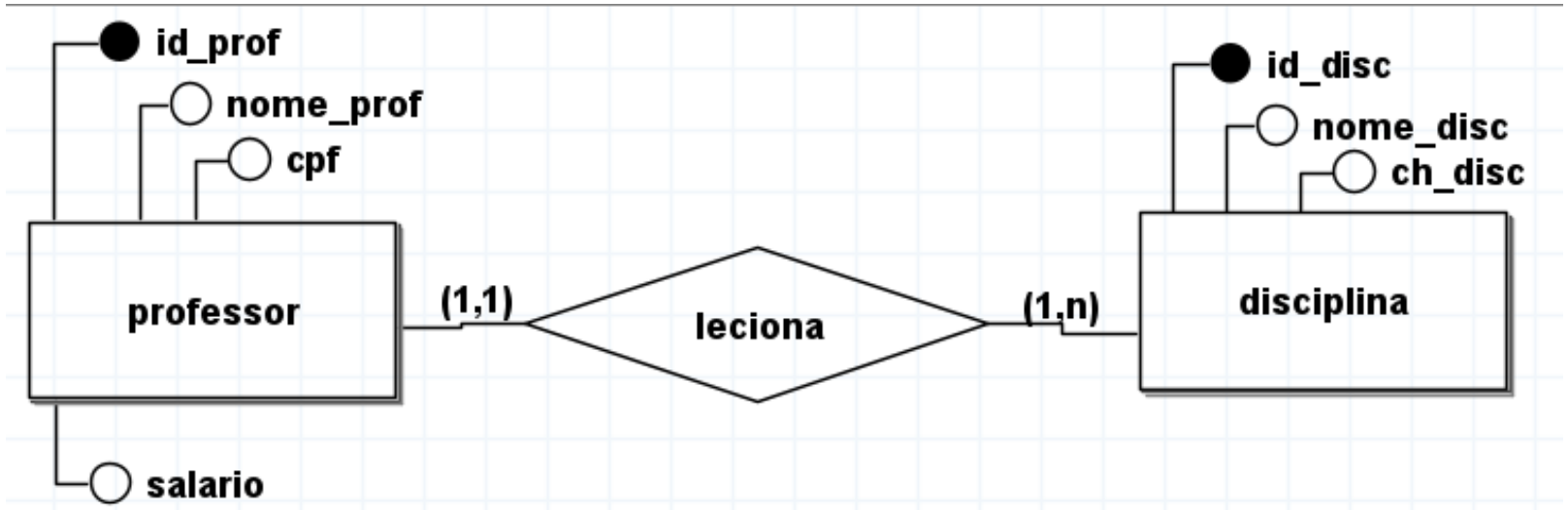


UPDATE

Id_prof	Nome_prof	cpf

Comandos - **EXERCÍCIO**

Crie as tabelas abaixo em SQL, e adicione pelo menos 5 registros para cada coluna.



Comandos DQL (Consulta)

- SELECT: É o comando mais importante do DQL.
  Ele serve para **buscar dados** de uma ou mais tabelas, podendo aplicar filtros, ordenações e junções.

- Exemplo:
  Select * from funcionário;
  -- Seleciona e exibe todas as colunas de funcionário

Comandos DQL (Consulta)

- Clausulas do SELECT:

Cláusula	Função principal
FROM	Define a tabela origem dos dados
WHERE	Filtra registros
GROUP BY	Agrupa resultados
HAVING	Filtra grupos
ORDER BY	Ordena resultados
JOIN / ON	Relaciona tabelas
LIMIT / TOP	Limita a quantidade de linhas retornadas

Comandos - **SELECT**

Seleciona todas(*) colunas da tabela professor

```
SELECT * FROM professor;
```

id_prof	nome_prof	cpf	salario
1	João Silva	12345678900	4500.00
2	Maria Oliveira	23456789011	5200.50
3	Carlos Santos	34567890122	4100.75
4	Ana Souza	45678901233	6000.00
5	Paulo Lima	56789012344	4800.25

O "" Seleciona tudo, se fosse necessário apenas nome_prof e cpf então seria:*

```
SELECT nome_prof, cpf FROM professor;
```

Comandos - **SELECT**

Seleciona as colunas distintas(não repetidas) da tabela professor

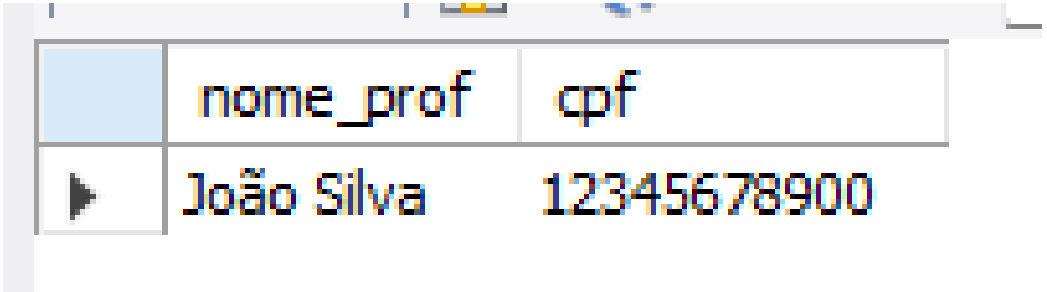
```
SELECT DISTINCT nome_prof FROM professor;
```

João Silva
Maria Oliveira
Ana Souza
Paulo Lima

Comandos - **SELECT**

Consulta com a restrição WHERE (onde), onde id_prof = 1

```
SELECT nome_prof, cpf
FROM professor
WHERE id_prof = 1;
```



nome_prof	cpf
João Silva	12345678900

Comandos - **SELECT**

Consulta com a restrição WHERE (onde) o salario esta entre 4000 e 5000

```
SELECT nome_prof, salario FROM professor
WHERE salario between 4000 and 5000;
```

	nome_prof	salario
▶	João Silva	4500.00
	Maria Oliveira	4100.75
	Paulo Lima	4800.25

Comandos - **SELECT**

Consulta com a restrição WHERE (onde) onde o salário é maior que 4000 e menor que 6000

```
SELECT nome_prof, salario FROM professor
WHERE salario > 4000 and salario <6000;
```

	nome_prof	salario
▶	João Silva	4500.00
	Maria Oliveira	5200.50
	Maria Oliveira	4100.75
	Paulo Lima	4800.25

Comandos - **SELECT**

Retorna nome e salario de professores que iniciam nome com a letra m.

```
SELECT nome_prof, salario FROM professor
WHERE nome_prof like ‘m%’;
```

	nome_prof	salario
▶	Maria Oliveira	5200.50
	Maria Oliveira	4100.75

Comandos - **SELECT**

Mostra os professores em ordem alfabética.(ASC = crescente, DESC = decrescente).

```
SELECT nome_prof, cpf
FROM Professor
ORDER BY nome_prof ASC;
```

nome_prof	cpf
Ana Souza	45678901233
João Silva	12345678900
Maria Oliveira	23456789011
Maria Oliveira	34567890122
Paulo Lima	56789012344

Apelidando tabelas com “as”

Mostra uma nova coluna com o salario aumentado em 10% apelidada de Atualizada.

```
select nome_prof, salario, salario*1.1 as "Atualizado"
from professor;
```

Apelidando tabelas com “as”

Cria um apelido para as colunas nome_prof e cpf.

```
SELECT
    nome_prof AS Professor,
    cpf AS Documento
FROM professor;
```

	Professor	Documento
▶	João Silva	12345678900
	Maria Oliveira	23456789011
	Maria Oliveira	34567890122
	Ana Souza	45678901233
	Paulo Lima	56789012344

Apelidando tabelas com “as”

Cria um apelido para a tabela professor, chamando de p.

```
SELECT p.nome_prof, p.cpf
FROM professor AS p;
```

nome_prof	cpf
João Silva	12345678900
Maria Oliveira	23456789011
Maria Oliveira	34567890122
Ana Souza	45678901233
Paulo Lima	56789012344

FUNÇÕES

UPPER

A função UPPER exibe o conteúdo da coluna em maiúscula.

```
SELECT upper(nome_prof), cpf FROM Professor;
```

	upper(nome_prof)	cpf
▶	JOÃO SILVA	12345678900
	MARIA OLIVEIRA	23456789011
	MARIA OLIVEIRA	34567890122
	ANA SOUZA	45678901233
	PAULO LIMA	56789012344

FUNÇÕES DE AGREGAÇÃO

Funções de Agregação: executam cálculos sobre uma coluna de valores.

- **SUM()**
- **MAX()**
- **MIN()**
- **AVG()**
- **COUNT()**

FUNÇÕES

SUM

Exibe a soma dos salários da tabela professor.

```
SELECT SUM(salario) AS Total_Salarios FROM professor;
```

	Total_Salarios
▶	24601.50

FUNÇÕES

MAX

Exibe o maior salário da tabela professor.

```
SELECT MAX(salario) AS Maior_Salario
FROM professor;
```

	Maior_Salario
▶	6000.00

FUNÇÕES

MIN

Exibe o menor salário da tabela professor.

```
SELECT MIN(salario) AS Menor_Salario
FROM professor;
```

	Menor_Salario
▶	4100.75

FUNÇÕES

AVG

Exibe a média dos salários da tabela professor.

```
SELECT AVG(salario) AS Media_Salarial
FROM professor;
```

	Media_Salarial
▶	4920.300000

FUNÇÕES

COUNT

Exibe a média dos salários da tabela professor.

```
SELECT COUNT(*) AS Total_Professores
FROM professor;
```

	Total_Professores
▶	5

FUNÇÕES

Todas em um select

Exibe a média dos salários da tabela professor.

```
SELECT
    SUM(salario) AS Total,
    MAX(salario) AS Maior,
    MIN(salario) AS Menor,
    AVG(salario) AS Media,
    COUNT(*) AS Quantidade
FROM professor;
```

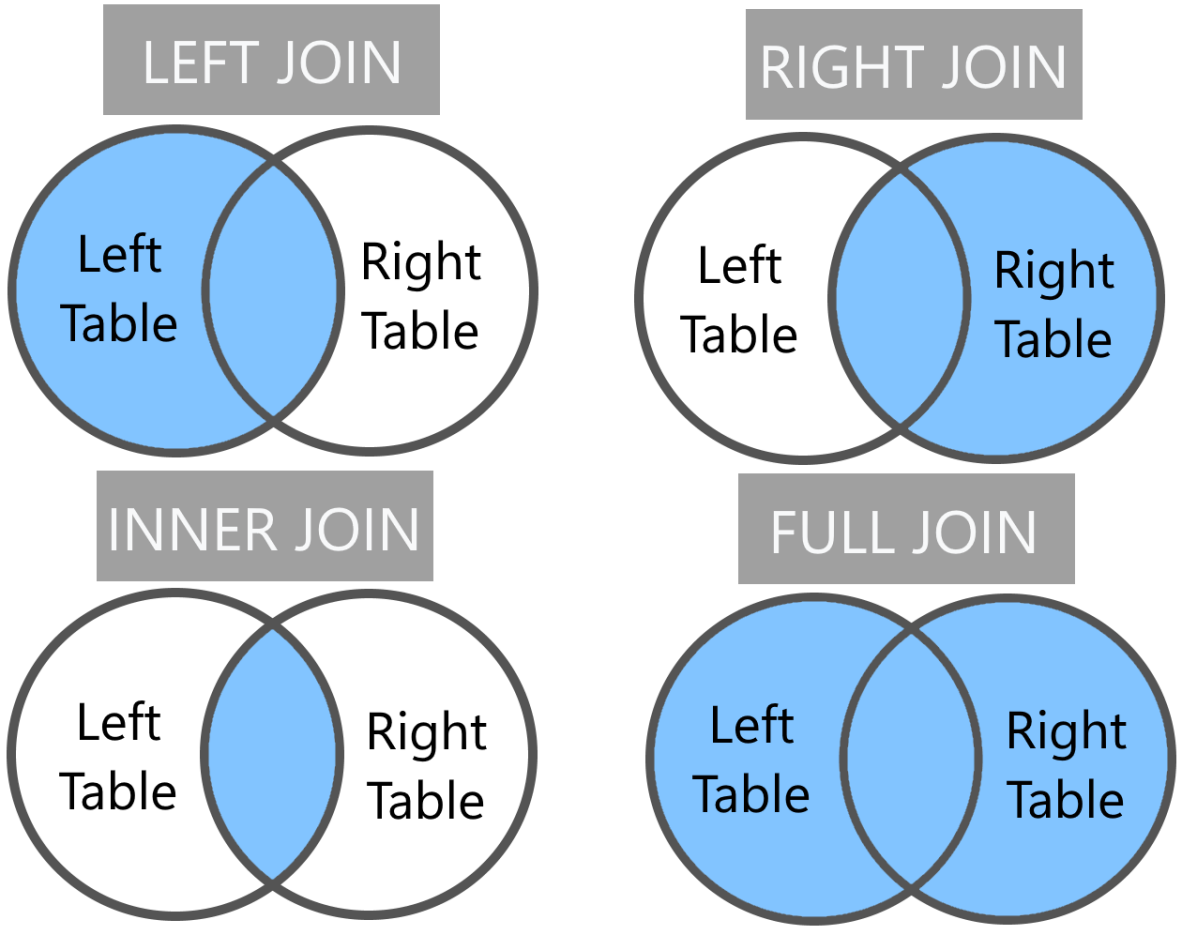
	Total	Maior	Menor	Media	Quantidade
▶	24601.50	6000.00	4100.75	4920.300000	5

JOINS (Junções)

Usado para unir dados de duas ou mais tabelas com base em uma coluna que elas têm em comum.

- **INNER JOIN (**Só quem tem relação)
- **LEFT JOIN (**Todos da tabela esquerda - com ou sem relação)
- **RIGHT JOIN (**Todos da tabela esquerda - com ou sem relação)
- **FULL JOIN (**Tudo de ambos os lados)

JOINS (Junções)



JOINS (Junções)

Usado para unir dados de duas ou mais tabelas com base em uma coluna que elas têm em comum.

- **INNER JOIN (**Só quem tem relação)
- **LEFT JOIN (**Todos da tabela esquerda - com ou sem relação)
- **RIGHT JOIN (**Todos da tabela esquerda - com ou sem relação)
- **FULL JOIN (**Tudo de ambos os lados)

JOINS (Junções)

INNER JOIN

- Retorna apenas os registros que **têm correspondência nas duas tabelas**

```
SELECT p.nome_prof,    d.nome_disc
FROM professor AS p
INNER JOIN disciplina AS d
ON p.id_prof = d.id_prof;
```

	nome_prof	nome_disc
▶	João Silva	Banco de Dados
	Maria Oliveira	Redes de Computadores
	Maria Oliveira	Programação em C
	Ana Souza	Segurança da Informação
	Paulo Lima	Sistemas Operacionais

JOINS (Junções)

INNER JOIN

- Retorna todos os registros da tabela da esquerda (professor)

```
SELECT p.nome_prof,    d.nome_disc
FROM professor as p
LEFT JOIN disciplina as d ON p.id_prof = d.id_prof;
```

nome_prof	nome_disc
João Silva	Banco de Dados
Maria Oliveira	Redes de Computadores
Maria Oliveira	Programação em C
Ana Souza	Segurança da Informação
Paulo Lima	Sistemas Operacionais
Joel	NULL

JOINS (Junções)

INNER JOIN

- Retorna todos os registros da tabela da direita (disciplina)

```
SELECT p.nome_prof,    d.nome_disc
FROM professor as p
RIGHT JOIN disciplina as d ON p.id_prof = d.id_prof;
```

nome_prof	nome_disc
João Silva	Banco de Dados
Maria Oliveira	Redes de Computadores
Maria Oliveira	Programação em C
Ana Souza	Segurança da Informação
Paulo Lima	Sistemas Operacionais
NULL	Algoritmo

JOINS (Junções)

FULL JOIN

- Retorna todos os registros das duas tabela.

```
SELECT p.nome_prof,    d.nome_disc
FROM professor p
LEFT JOIN disciplina d
ON p.id_prof = d.id_prof

UNION

SELECT     p.nome_prof,    d.nome_disc
FROM professor p
RIGHT JOIN disciplina d
 ON p.id_prof = d.id_prof;
```